DESY 99-155
ROM2F 99-37
hep-lat/9910024

SSOR preconditioning in simulations of the QCD Schrödinger functional

ALPHA Collaboration

Marco Guagnelli [a] and Jochen Heitger [b]

[a] *Dipartimento di Fisica, Università di Roma "Tor Vergata" and INFN, Sezione di Roma II*
Via della Ricerca Scientifica 1, I-00133 Rome, Italy

[b] *Deutsches Elektronen-Synchrotron, DESY Zeuthen*
Platanenallee 6, D-15738 Zeuthen, Germany

Abstract

We report on a parallelized implementation of SSOR preconditioning for O(a) improved lattice QCD with Schrödinger functional boundary conditions. Numerical simulations in the quenched approximation at parameters in the light quark mass region demonstrate that a performance gain of a factor ~ 1.5 over even-odd preconditioning can be achieved.

Key words: lattice QCD; O(a) improvement; Schrödinger functional; SSOR preconditioning; parallelization; quenched simulations

PACS: 11.15.Ha; 12.38.Gc; 2.60.Cb; 2.60.Dc

1 Introduction

One of the severe problems in lattice QCD from the practical point of view is the numerical inversion of sparse matrices. It nearly enters every Monte Carlo simulation, either in the quenched approximation of the theory in applying the inverse of the discretized Dirac operator on source vectors for the computation of quark propagators, or in full QCD with Hybrid Monte Carlo like algorithms, where a similar operation is necessary when calculating the fermionic force in order to include the quark field dynamics in the updating process of the gauge fields.

In recent years substantial progress has been made by the use of Krylov subspace iterative solvers in conjunction with preconditioning techniques. (See e.g. [1] for a textbook reference and the reviews [2–4], which contain an extensive bibliography.) Popular choices for the inverter to be mentioned in the context of lattice simulations are the Conjugate Gradient (CG) algorithm [5,6], the Minimal Residual (MR) algorithm [7,8] and above all the stabilized Bi-Conjugate Gradient (BiCGStab) algorithm [9,10]. The latter now seems commonly established as the most efficient solver in a vast majority of QCD applications [11,12], particularly in the parameter region of small quark masses.

In the area of preconditioning, any new (and potentially competitive) idea should be confronted with an even-odd (e/o) decomposition of the Dirac matrix [13,14], which both for ordinary Wilson fermions and together with an O(a) improvement term [15] has become the state-of-the-art method. An earlier step towards an alternative approach was the incomplete LU factorization [16] utilizing a *globally*-lexicographic ordering of the lattice points and thereby already related to SSOR. However, it turns out to be unsatisfactory when an implementation on grid-oriented parallel computers is envisaged [17]. Finally, the Wuppertal group invented a parallelization of symmetric successive overrelaxed (SSOR) preconditioning, a variant of the classical Gauss-Seidel iteration [1], resting upon a *locally*-lexicographic ordering of the points within a given (sub-)grid of the total space-time lattice [18–22]. They showed that at least for Wilson fermions an SSOR preconditioner can perform much better that the standard e/o one.

Since many comparative studies of the properties of the various inversion algorithms and preconditioning methods are already available in the literature [4,12,17], we restrict ourselves in the sequel exclusively to the case of the O(a) improved Wilson-Dirac operator involving the Sheikholeslami-Wohlert clover term [23] within the Schrödinger functional of QCD [24,25].

The *Schrödinger functional (SF)* is defined as the partition function of QCD in a space-time cylinder of extension $L^3 \times T$ with periodic boundary conditions

in the space directions and (inhomogeneous) Dirichlet boundary conditions at times 0 and T. As detailed e.g. in refs. [25,26], the SF has proven to be a valuable tool for computing the running coupling constant in a finite-size scaling analysis as well as for extracting phenomenological quantities from simulations in physically large volumes [27,28]. On the other hand, the $O(a)$ *improved Wilson-Dirac operator* is now a good candidate to probe continuum QCD by means of numerical simulations on the lattice: most coefficients multiplying the counterterms required for a complete removal of the leading $\mathrm{O}(a)$ lattice artifacts of action and quark currents are known non-perturbatively in the quenched approximation [25,29,30], and also for the action in the case of two flavours of dynamical quarks [31]. Therefore, a quantitative investigation of the performance of a (parallelized) SSOR preconditioner for the SF setup incorporating $\mathrm{O}(a)$ improvement appears to be of natural interest. This is what we intend with the present communication.

One might ask what should be different from periodic boundary conditions in space *and* time. At first, it is in principle not excluded that with Dirichlet boundary conditions a (specifically preconditioned) solver has generally lower iteration numbers. Another point concerns a definite advantage in the actual implementation, since the SF allows to circumvent inefficient conditional statements as will be explained later.

2 SSOR preconditioning

The basic problem posed is to solve a system of linear equations of the type

$$\mathcal{A}X = Y \quad \Leftrightarrow \quad R \equiv \mathcal{A}X - Y = 0\,. \tag{1}$$

To fix some notation, small Greek letters denote scalars, capital Latin ones vectors with components in small letters, and matrices have calligraphic symbols; $(X, Y) = \sum_i x_i^* y_i$ is the usual scalar product. In lattice QCD, $\mathcal{A}$ represents the discretized Dirac operator, a huge sparse matrix of rank $\Omega \times 4 \times 3$ emerging from the nearest neighbour couplings of the quark and gluon field variables in position space after discretization of the interacting continuum theory. More precisely, Ω is the volume of the four-dimensional space-time lattice, and the latter factors are connected to Dirac and SU(3) colour spaces. Physically, the solution of eq. (1) can be regarded as a fermionic Green function (i.e. a quark propagator).

SSOR preconditioning consists of solving the system

$$\mathcal{V}_1^{-1}\mathcal{A}\mathcal{V}_2^{-1}\tilde{X} = \tilde{Y}\,, \quad \tilde{X} = \mathcal{V}_2 X\,, \quad \tilde{Y} = \mathcal{V}_1^{-1} Y \tag{2}$$

instead of $\mathcal{A}X = Y$, where in the present context $\mathcal{A}$ stands for $\hat{\mathcal{Q}}$, the lattice Dirac operator of Wilson fermions supplemented with a local O(a) improvement term $a\sigma_{\mu\nu}F_{\mu\nu}(x)$, which is composed of the SU(3)–valued gauge link variables $U(x,\mu)$ to form a clover leaf [23]:

$$\begin{aligned}\hat{\mathcal{Q}} = {} & \mathbf{1}\delta_{x,y} \\ & - \kappa \sum_{\mu} \Big\{ (\mathbf{1}-\gamma_\mu) U(x,\mu)\delta_{x,y-a\hat{\mu}} + (\mathbf{1}+\gamma_\mu) U(x-a\hat{\mu},\mu)^+ \delta_{x,y+a\hat{\mu}} \Big\} \\ & + c_{\rm sw}\, \frac{i}{2}\, a\, \kappa \sum_{\mu,\nu} \sigma_{\mu\nu} F_{\mu\nu}(x)\, \delta_{x,y} \end{aligned} \tag{3}$$

with $F_{\mu\nu}(x)$ equal to

$$\begin{aligned} \frac{1}{8} \Big\{ & \Big[U(x,\mu)U(x+a\hat{\mu},\nu)U(x+a\hat{\nu},\mu)^+ U(x,\nu)^+ \\ & + U(x,\nu)U(x+a\hat{\nu}-a\hat{\mu},\mu)^+ U(x-a\hat{\mu},\nu)^+ U(x-a\hat{\mu},\mu) \\ & + U(x-a\hat{\mu},\mu)^+ U(x-a\hat{\nu}-a\hat{\mu},\nu)^+ U(x-a\hat{\nu}-a\hat{\mu},\mu) U(x-a\hat{\nu},\nu) \\ & + U(x-a\hat{\nu},\nu)^+ U(x-a\hat{\nu},\mu) U(x-a\hat{\nu}+a\hat{\mu},\nu) U(x,\mu)^+ \Big] \\ & - \Big[\cdots \Big]^+ \Big\}. \end{aligned} \tag{4}$$

$c_{\rm sw}$ is an improvement coefficient non-perturbatively determined in quenched and two-flavour QCD [29,31]. (The slight modification to (3) induced by the Dirichlet boundary conditions in time direction are not written out here.) If one introduces the decomposition [1]

$$\hat{\mathcal{Q}} = \mathcal{D} - \mathcal{L} - \mathcal{U} \tag{5}$$

into block diagonal, block lower-triangular and block upper-triangular parts with respect to position space, the SSOR preconditioner is defined in terms of these matrices and a non-zero relaxation parameter $\omega_{\rm SSOR}$ (which serves to reduce the iteration number) through the choice

$$\mathcal{V}_1 = \Big(\frac{1}{\omega_{\rm SSOR}}\, \mathcal{D} - \mathcal{L} \Big) \Big(\frac{1}{\omega_{\rm SSOR}}\, \mathcal{D} \Big)^{-1}, \quad \mathcal{V}_2 = \frac{1}{\omega_{\rm SSOR}}\, \mathcal{D} - \mathcal{U}\,. \tag{6}$$

Now it is advantageous to exploit in (2) the so-called 'Eisenstat trick' [32], embodied in the identity

$$\frac{1}{\omega_{\rm SSOR}}\, \mathcal{D} \Big(\frac{1}{\omega_{\rm SSOR}}\, \mathcal{D} - \mathcal{L} \Big)^{-1} (\mathcal{D} - \mathcal{L} - \mathcal{U}) \Big(\frac{1}{\omega_{\rm SSOR}}\, \mathcal{D} - \mathcal{U} \Big)^{-1}$$

[1] The case of Wilson-Dirac fermions is recovered by setting $\mathcal{D}$ to the unit matrix.

$$= \left(\mathbf{1} - \omega_{\mathrm{SSOR}}\mathcal{L}\mathcal{D}^{-1}\right)^{-1}\left\{\mathbf{1} + (\omega_{\mathrm{SSOR}} - 2)\left(\mathbf{1} - \omega_{\mathrm{SSOR}}\mathcal{U}\mathcal{D}^{-1}\right)^{-1}\right\} + \left(\mathbf{1} - \omega_{\mathrm{SSOR}}\mathcal{U}\mathcal{D}^{-1}\right)^{-1}, \quad (7)$$

to save on computational costs: namely, it implies that any matrix-vector product with the preconditioned matrix $\mathcal{V}_1^{-1}\hat{\mathcal{Q}}\mathcal{V}_2^{-1}$ essentially gives rise to a backward substitution and a subsequent forward substitution process, corresponding to the application of the (non-block) upper and lower triangular matrices $\mathbf{1} - \omega_{\mathrm{SSOR}}\mathcal{U}\mathcal{D}^{-1}$ and $\mathbf{1} - \omega_{\mathrm{SSOR}}\mathcal{L}\mathcal{D}^{-1}$, respectively. These relations reflect that SSOR splits any application of $\hat{\mathcal{Q}}$, which usually enters the inversion procedure associated with (1), into an equivalent but much simpler sequence of arithmetic operations with the set of matrices $\{\mathcal{D}, \mathcal{L}, \mathcal{U}\}$.

Then, combining the BiCGStab algorithm with SSOR preconditioning, an iterative numerical solution of the system in eq. (1) up to a given precision ϵ is obtained by the prescription [2] (preconditioned quantities carry a tilde):

initialization with guess X_0 :

$$R_0 = Y - \hat{\mathcal{Q}}X_0$$
$$(\mathbf{1} - \omega_{\mathrm{SSOR}}\mathcal{L}\mathcal{D}^{-1})\tilde{R}_0 = R_0$$

$$\tilde{\mathrm{R}} \equiv \tilde{R}_0 \quad \rho_0 \equiv 1$$
$$\tilde{V}_0 = \tilde{P}_0 \equiv 0 \quad \alpha_0 = \omega_0 \equiv 1$$

k-th iteration ($k \geq 1$) :

$$\rho_k = (\tilde{\mathrm{R}}, \tilde{R}_{k-1})$$

$$\beta = \frac{\rho_k \alpha_{k-1}}{\rho_{k-1}\omega_{k-1}}$$

$$\tilde{P}_k = \tilde{R}_{k-1} + \beta\tilde{P}_{k-1} - \beta\omega_{k-1}\tilde{V}_{k-1}$$

$$(\mathbf{1} - \omega_{\mathrm{SSOR}}\mathcal{U}\mathcal{D}^{-1})W = \tilde{P}_k\,, \quad Z' = \tilde{P}_k + (\omega_{\mathrm{SSOR}} - 2)W$$
$$(\mathbf{1} - \omega_{\mathrm{SSOR}}\mathcal{L}\mathcal{D}^{-1})Z = Z'\,, \quad \tilde{V}_k = W + Z$$

$$\alpha_k = \frac{\rho_k}{(\tilde{\mathrm{R}}, \tilde{V}_k)}$$

$$\tilde{S} = \tilde{R}_{k-1} - \alpha_k\tilde{V}_k$$

[2] The algorithm can straightforwardly formulated for other solvers, like MR for instance. Note, however, that in the chiral quark mass regime MR is generally of worse performance.

$$(\mathbf{1} - \omega_{\mathrm{SSOR}}\mathcal{U}\mathcal{D}^{-1})U = \tilde{S}\,, \quad Z' = \tilde{S} + (\omega_{\mathrm{SSOR}} - 2)U$$
$$(\mathbf{1} - \omega_{\mathrm{SSOR}}\mathcal{L}\mathcal{D}^{-1})Z = Z'\,, \quad \tilde{T} = U + Z$$

$$\omega_k = \frac{(\tilde{T}, \tilde{S})}{(\tilde{T}, \tilde{T})}$$

$$\tilde{X}_k = \tilde{X}_{k-1} + \omega_k U + \alpha_k W$$

$$\tilde{R}_k = \tilde{S} - \omega_k \tilde{T}$$

The stopping criterion to be imposed on the preconditioned residual is

$$\frac{(\tilde{R}_k, \tilde{R}_k)}{(\tilde{X}_k, \tilde{X}_k)} \leq \epsilon^2\,. \tag{8}$$

Some comments are in order.

- Taking the forward substitution as example, one calculates recursively in terms of the (block) components Z_i, Z'_i, $\mathcal{D}_{ij}$ and $\mathcal{L}_{ij}$ of Z, Z', $\mathcal{D}$ and $\mathcal{L}$:

$$\forall\, i: \quad Z_i = Z'_i + \sum_{j=1}^{i-1} \mathcal{L}_{ij} H_j\,, \quad H_j = \omega_{\mathrm{SSOR}} \left(\mathcal{D}^{-1}\right)_{jj} Z_j \Big|_{j \leq i-1}\,. \tag{9}$$

  I.e. thanks to the triangularity of $\mathcal{L}$ (and $\mathcal{U}$) it can be done economically without involving a plain matrix multiplication, and owing to the sparsity of $\hat{\mathcal{Q}}$ only a few j actually contribute to the sum. Provided that the inverses $(\mathcal{D}^{-1})_{ii}$ are pre-computed, the backward and forward solves together are exactly as expensive as one application of the whole matrix $\hat{\mathcal{Q}}$ plus one additional $\mathcal{D}$ multiplication.
- In contrast to the unimproved case (where $X_k = \tilde{X}_k$ follows immediately), the solution of the original system $\hat{\mathcal{Q}}X = Y$ is now $X_k = \omega_{\mathrm{SSOR}}^{-1}\mathcal{D}^{-1}\tilde{X}_k$.
- Choose $X_0 = 0 = \tilde{X}_0$ as initial guess for the solution to avoid an application of $\mathcal{D}$, as part of $\hat{\mathcal{Q}}$, at the beginning, which yields $R_0 = Y - \hat{\mathcal{Q}}X_0 = Y$. If $\mathcal{D}$ is not needed elsewhere, this might be favourable in view of possible memory limitations of the hardware (e.g. setting an upper bound on the accessible lattice volumes), because we made the experience that savings in iteration number when using an available solution of a foregoing inversion as initial guess are generically negligible.
- Since in order to save on computational cost the stopping criterion (in our runs $\epsilon^2 = 10^{-12}, 10^{-13}$) is conveniently based on $\tilde{R}_k$, one might want to re-compute R_k at the end to test for convergence again and eventually — if the solution is not yet accurate enough — to continue the iteration with a stronger stopping criterion imposed on $\tilde{R}_k$.

- In the SSOR scheme the minimal number of vectors to be stored at each iteration k is 9, if $\tilde{R}_{k-1}$ and $\tilde{S}$ share the same memory location, and if the source vector Y is overwritten by the iterative solutions $\tilde{X}_k$.
- Due to the overall Dirac and colour structure of the fermion field variables at every lattice point, $\hat{\mathcal{Q}}$ is genuinely partitioned into blocks of dimension 12×12. The dependence of the speed of the SSOR preconditioner on the diagonal (sub-)block size of $\mathcal{D}_{ij}$ was not studied.

2.1 Implementation for the Schrödinger functional

For the implementation of SSOR preconditioning on a parallel computer, the ordering of the lattice points plays a key rôle [3]. It determines the shape of $\hat{\mathcal{Q}}$ via the pattern of its non-zero entries and thereby the degree of parallelism and the efficiency of the preconditioner. Via adapting a locally-lexicographic scheme we closely follow refs. [19,21], where different orderings and their consequences on the parallelization have been discussed. Hence we only describe the salient issues characteristic for the SF approach.

Assume that a given space-time lattice is matched on the, say, three-dimensional grid of processing nodes of a parallel computer. Then each node occupies a local lattice, where three of its extensions are ratios of the total lattice sizes in the respective directions and the corresponding numbers of processors. The locally-lexicographic ordering ('colouring') is ensured by an alphabetic ordering of the lattice sites belonging to these local lattices. Associating a colour with each fixed position within the local lattices, the original lattice is divided into groups, whose members couple to sublattice points of different colours only. In this way it becomes obvious that the forward and backward substitutions, e.g. to get Z_i as in eq. (9), can be handled in parallel within the coloured groups, since for all positions of a given colour only their predecessors — in the lexicographic sense — enter the necessary computations [4]. Because among these lexicographically preceding sites, lattice points living on the neighbouring processors contribute too, the quantities H_i of (9) will have to be communicated from those processing nodes to the site in question.

At this point we have to note that there is a crucial difference between a situation with ordinary (anti-)periodic boundary conditions in all four space-time directions and our SF setup with Dirichlet boundary conditions in time. To see this difference in more detail, let us resort to an example in one dimension. The four points in figure 1 define a one-dimensional lattice with periodic

[3] We remark that traditional e/o preconditioning can be interpreted as SSOR preconditioning of the even-odd ordered system [19].

[4] So the general strategy would be to maximize the number of coloured groups, while maintaining its strengths in accordance with the desired parallelization.

boundary conditions, and one can think of a 'coupling matrix' between the sites of mutual dependence. Since only nearest neighbour sites are coupled, we

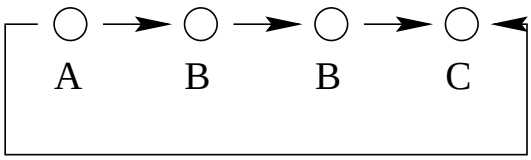

Fig. 1. *A one-dimensional example. The direction of the arrows shows the data flow: sites of kind* **B** *need information from site* **A***, and so forth.*

immediately see that in applying eq. (9) one has to distinguish between three cases. (We restrict to the forward solve, because it is straightforward to work out the necessary changes for the backward solve.)

A The point is on the 'left', i.e. it has coordinate $x = 1$: (9) simply becomes

$$Z_1 = Z'_1 \,. \tag{10}$$

B The point lies within the 'bulk', i.e. it has coordinate $x = 2, 3$: (9) becomes

$$Z_x = Z'_x + \mathcal{L}_{x,x-1} H_{x-1} \,, \quad x = 2, 3 \,. \tag{11}$$

C The point is on the 'right', i.e. it has coordinate $x = 4$: then (9) becomes

$$Z_4 = Z'_4 + \mathcal{L}_{4,3} H_3 + \mathcal{L}_{4,1} H_1 \,. \tag{12}$$

It is clear that we have those three cases for each dimension, in which either periodic or antiperiodic boundary conditions are prescribed. For a four-dimensional lattice this leads to $3^4 = 81$ different cases to be handled, and it is natural to implement the algorithm with a `do`–loop over all the lattice points and some `if`–statements to discriminate between the 81 cases. The *parallel* version of the algorithm just described needs minor modifications: looking at figure 2, only the case **C** has to be replaced with

$$Z_4 = Z'_4 + \mathcal{L}_{4,3} H_3 + \mathcal{L}_{4,1} H_1^{(\text{next processor})} \,, \tag{13}$$

where now the coordinate index has a local meaning, labelling the sites inside the sublattice residing on a given processor, and also the processor's mesh is assumed to have periodic boundary conditions in the sense that the processor 'next' to the last one in each direction (rightmost in figure 2) is to be identified with the first one in the same direction (leftmost in figure 2).

Our numerical simulations were done on the 8 – 512 nodes APE-100 massively parallel computers with cubic topology and nearest neighbour communication, built up of an array of elementary processing boards with $2 \times 2 \times 2$ nodes

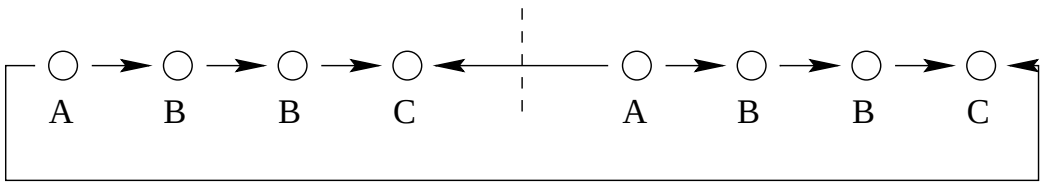

Fig. 2. *A one-dimensional parallel example. As in the non-parallel case, figure 1, the arrows represent the data flow involved.*

[33,34]. They possess a SIMD (single-instruction multiple-data) architecture and are either suited to distribute a single, typically large, lattice over the whole machine or to simulate in parallel several independent copies of a lattice on subsets of the machine in the case of smaller volumes.

With such a cubic topology it suggests itself to keep the whole time extent of the lattice within the processors, and to only split the spacelike volume into sublattice fractions with respect to the three-dimensional processor grid. If *SF boundary conditions* are adopted, one can make sure that, as far as the time direction with its first and last timeslices fixed to the boundary values is concerned, a site belongs always to the 'bulk'. This lowers the number of cases to be distinguished in order to implement the forward and backward solves to $3^3 = 27$. So it becomes rather near at hand to encode these cases explicitly in a sensible arrangement of nested `do`–loops alone, i.e. via an outer loop over the full time coordinate and inner ones over the coordinates of the local space directions on every processor to exhaust the total lattice volume; thereby the usage of any `if`–statements is completely avoided. Here arises the significant advantage of our implementation: the latter type of statements cause — especially so on the APE-100 machines — a drastic deterioration of the performance by breaking the so-called 'optimization blocks' recognized by the compiler. In practice, the contents of the registers is lost each time a branching statement like `if`, `do`, `call subroutine`, ... is encountered, because this amounts to a break in the memory pipeline pre-loaded before. Writing out the 27 distinct cases explicitly, however, reduces the impact of the largest part of such statements, so that finally we are able to arrive at a substantial speed-up of the code [5].

2.2 Performance tests

After realizing the implementation outlined above, we first have investigated the influence of the relaxation parameter ω_{SSOR} on the numerical solution pro-

[5] Of course one can imagine to write down analogously the 81 different cases for the familiar periodic situation [19], but then the size of the executable file might easily exceed the integer and/or program memory limits of the machine.

cedure for the linear system in eq. (1). It is quantified through the ratio of the numbers of iterations to solve the system with the e/o preconditioned matrix, $N_{\rm e/o}$, and with the SSOR preconditioned one, $N_{\rm SSOR}$, under otherwise identical conditions. We show these ratios in dependence on $\omega_{\rm SSOR}$, averaged over a set of O(10) propagator computations occurring in quenched simulations, for two lattice sizes with SF boundary conditions in figure 3. Upon distributing these

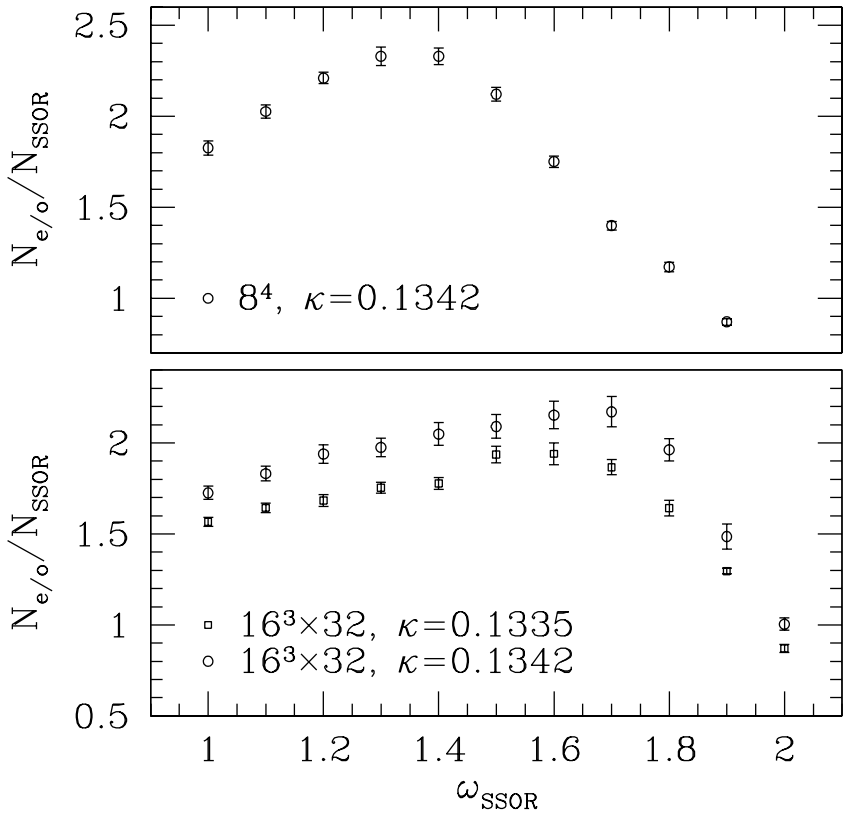

Fig. 3. *Improvement factor in the number of BiCGStab iterations, when simulating with SSOR and different choices for* $\omega_{\rm SSOR}$ *instead of e/o preconditioning. The lattice sizes are* 8^4 *and* $16^3\times 32$*, with parameters* $\beta = 6.0$ *and* $\kappa = 0.1335, 0.1342$*.*

lattices over the meshes of nodes of the APEs, the lattice extensions per node were $4^3\times 8$ and $2^3\times 32$. The course of $N_{\rm e/o}/N_{\rm SSOR}$ confirms a conclusion of the Wuppertal group [21] that between $\omega_{\rm SSOR}\simeq 1.3$ and $\omega_{\rm SSOR}\simeq 1.6$ the gain in the number of iterations needed for the fermion matrix inversions reaches its maximum: the corresponding improvement factor is around or even above 2, while the tendency for a further increase of $N_{\rm e/o}/N_{\rm SSOR}$ towards the chiral limit (i.e. larger κ and smaller quark mass) is also seen. Above $\omega_{\rm SSOR}\simeq 1.6-1.8$ this ratio drops rapidly and the gain gets lost; therefore, we take over $\omega_{\rm SSOR}=1.4$ to be considered as an 'optimal' compromise irrespective of the definite values for lattice sizes and/or parameters.

Now we pass to the central question, whether the gain in the number of iterations also translates into a visible CPU time gain. Of course, as already pointed out previously, this will depend on the hardware architecture of the machine in use as well as on the individual implementation. At the peak in the upper diagram of figure 3, for instance, a total saving of 1.5 in the spent simulation time can be attained. In table 1 we collect the approximate performance gain factors in units of iteration number and net CPU time, which were found in the situation of a realistic (quenched) QCD simulation in physically large volumes. Here we applied SSOR and e/o preconditioning for a

Table 1
Approximate gain factors of SSOR over e/o in iteration number and net CPU time

sublattice/node	subvolume/node	iteration gain	performance gain
$2^3 \times 32$	$8 \cdot 32$	2.0	1.4
$2^2 \times 4 \times 32$	$16 \cdot 32$	2.1	1.4
$2 \times 4^2 \times 32$	$32 \cdot 32$	2.2	1.5

set of 300 fermion matrix inversions on thermalized $16^3 \times 32$ configurations at $\beta = 6.0$ and $\kappa = 0.1335, 0.1342$, where the relaxation parameter was set to $\omega_{\mathrm{SSOR}} = 1.4$ throughout and the pseudoscalar mass at those couplings is $am_{\mathrm{PS}} = 0.388, 0.300$ [27,28]. Moreover, we examined the dependence of the SSOR preconditioner on the fractional grid size per node treated by a single processor. As illustrated by the numbers in table 1 and in figure 4, the iteration number ratio and thus the preconditioning efficiency slightly increases with growing volumes of the different local subgrids, $2^3 \times 32$, $2^2 \times 4 \times 32$ and $2 \times 4^2 \times 32$, if the $16^3 \times 32$ lattice is spread over the 512, 256 and 128 processors of the available machines, respectively. This complies with the heuristic expectation that an enlarged number of coloured groups, i.e. sets of points at the same fixed position within the local sublattices (according to their locally-lexicographic ordering), entails a measurable iteration gain in the inverter, whereas the performance stays nearly unchanged owing to less parallelism and hence a small accompanying inter-processor communication overhead. Because of the equivalence of e/o preconditioning to a colouring into two groups assigned to the even and odd sublattice, one can accommodate the e/o iteration number in the upper part of figure 4 too. Additionally, the points in both diagrams indicate once more the even better behaviour of BiCGStab-SSOR in the range of lighter quark masses.

The foregoing observations are supported by the large scale simulations in the strange quark mass region underlying the extraction of hadron masses and matrix elements in quenched QCD with the SF reported in refs. [27,28].

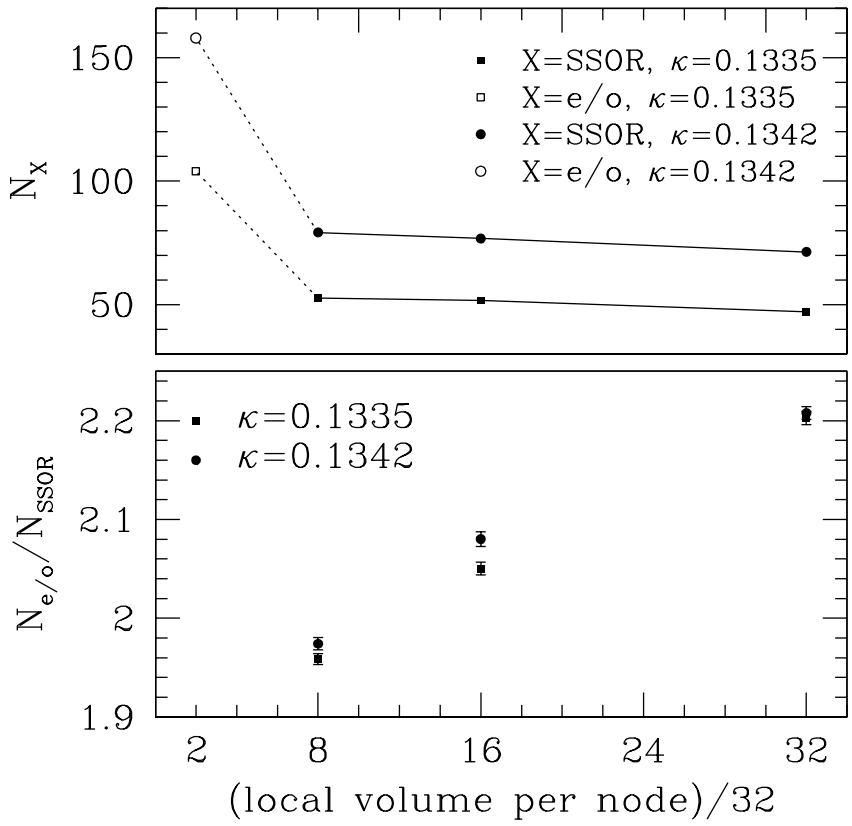

Fig. 4. *Upper part: BiCGStab iteration number in dependence of the local lattice volume per node after distributing a* $16^3 \times 32$ *lattice on different three-dimensional meshes of processing nodes. Note that the common time extent of the local lattices,* $T = 32$*, has been divided out. The e/o iteration numbers are included for comparison, and the lines are only meant to guide the eye. Lower part: Associated improvement factors in the number of iterations.*

There, at $\beta = 6.1, 6.2$ on $24^3 \times T$ lattices (with $3^3 \times T$ sublattices per node and $T = 40, 48$) and at $\beta = 6.45$ on a $32^3 \times 64$ lattice (with $4^2 \times 8 \times 64$ sublattice per node), SSOR enabled to save against e/o always CPU time factors of around $1.5 - 1.6$.

Altogether these results clearly reveal that the replacement of e/o by SSOR preconditioning to solve the O(a) improved Wilson-Dirac equation in the SF scheme pays off in real simulation costs. In contrast to what the authors in [21] obtain, for the same local diagonal block size of $\hat{Q}$ and computer class, from QCD simulations including the clover term with ordinary boundary conditions, the SF type of boundary conditions allow for an efficient implementation of SSOR preconditioning also on massively parallel machines with an architecture equal or similar to that of the APE-Quadrics systems.

3 Conclusion

We have demonstrated in numerical simulations of quenched lattice QCD with the Sheikholeslami-Wohlert quark action that the increase of performance between even-odd and SSOR preconditioning in a parallel implementation can be a factor ~ 1.5, when Schrödinger functional boundary conditions are employed.

Opposed to the more standard situation with periodic boundary conditions in all directions studied in ref. [21], the gain factors in real time consumption come out to be significantly better here. The main reason for this originates in the lower number of cases (27 versus 81) to be distinguished explicitly, when the contributions to a given site among the locally-lexicographically ordered points of the local sublattices residing on the processors have to be collected: the avoidance of any conditional statements in the solver routines evades unwanted breaks in the data flow within the registers of the computer, which then directly translates into a considerable gain in units of CPU time. We have to emphasize that this inherent sensitivity to pipeline optimization might be — at least partly — a special feature of the APE-100 environment. Nevertheless, since some conclusions drawn from the investigations in [20–22] refer to the identical particular machines, our findings should be of interest in the same context and can be compared with the results stated there.

As the Schrödinger functional formulation of QCD is physically already well accepted to be a viable framework to address many problems in the non-perturbative low energy regime of the theory [25,28], the observed evidence for a performance gain of SSOR (together with BiCGStab as the inverter) for the O(a) improved Wilson-Dirac operator in actual run time — also on a parallel computer — constitutes a further benefit of this scheme. Therefore, the feasibility of an efficient implementation of this preconditioner does not only provide an important algorithmic information by itself, but even more is also promising for any kind of future applications of the Schrödinger functional in lattice QCD.

Acknowledgements

This work is part of the ALPHA collaboration research programme. We thank DESY for allocating computer time on the APE-Quadrics computers at DESY Zeuthen to this project and the staff of the computer centre at Zeuthen for their support. We are grateful to Rainer Sommer for discussions, useful suggestions and a critical reading of the manuscript. J. H. also thanks Burkhard Bunk and Andreas Hoferichter for discussions.

References

[1] G.H. Golub and C.F.V. Loan, Matrix Computations, 3rd ed. (The Johns Hopkins University Press, Baltimore, 1996).

[2] R. Freund, G.H. Golub and N. Nachtigal, Acta Numerica (1992) 57.

[3] R. Barrett et al., Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods (SIAM, Philadelphia, PA, 1994).

[4] A. Frommer, Nucl. Phys. Proc. Suppl. B53 (1997) 120, hep-lat/9608066.

[5] M.R. Hestenes and E. Stiefel, J. Res. Nat. Bur. Stand. 49 (1952) 409.

[6] V. Faber and T. Manteuffel, SIAM J. Numer. Anal. 21 (1984) 352.

[7] S.C. Eisenstat, H.C. Elman and M.H. Schultz, SIAM J. Numer. Anal. 20 (1983) 345.

[8] P. Rossi, C.T.H. Davies and G.P. Lepage, Nucl. Phys. B297 (1988) 287.

[9] H. van der Vorst, SIAM J. Sci. Stat. Comput. 13 (1992) 631.

[10] M.H. Gutknecht, SIAM J. Sci. Stat. Comput. 14 (1993) 1020.

[11] A. Frommer, V. Hannemann, B. Nöckel, T. Lippert and K. Schilling, Int. J. Mod. Phys. C5 (1994) 1073, hep-lat/9404013.

[12] G. Cella, A. Hoferichter, V.K. Mitrjushkin, M. Müller-Preussker and A. Vicere, Int. J. Mod. Phys. C7 (1996) 787, hep-lat/9606003.

[13] T.A. DeGrand, Comput. Phys. Commun. 52 (1988) 161.

[14] T.A. DeGrand and P. Rossi, Comput. Phys. Commun. 60 (1990) 211.

[15] K. Jansen and C. Liu, Comput. Phys. Commun. 99 (1997) 221, hep-lat/9603008.

[16] Y. Oyanagi, Comput. Phys. Commun. 42 (1986) 333.

[17] G. Hockney, Nucl. Phys. Proc. Suppl. B17 (1990) 301.

[18] U. Block, A. Frommer and G. Mayer, Parallel Computing 14 (1990) 61.

[19] S. Fischer et al., Comput. Phys. Commun. 98 (1996) 20, hep-lat/9602019.

[20] N. Eicker et al., Nucl. Phys. Proc. Suppl. B63 (1998) 955, hep-lat/9709143.

[21] W. Bietenholz et al., Comput. Phys. Commun. 119 (1999) 1, hep-lat/9807013.

[22] N. Eicker et al., Nucl. Phys. Proc. Suppl. B73 (1999) 850, hep-lat/9809038.

[23] B. Sheikholeslami and R. Wohlert, Nucl. Phys. B259 (1985) 572.

[24] M. Lüscher, R. Narayanan, P. Weisz and U. Wolff, Nucl. Phys. B384 (1992) 168, hep-lat/9207009.

[25] R. Sommer, Non-perturbative Renormalization of QCD, Lectures given at 36th Internationale Universitätswochen für Kernphysik und Teilchenphysik, Schladming, Austria, 1 – 8 Mar 1997, hep-ph/9711243.

[26] M. Lüscher, R. Sommer, P. Weisz and U. Wolff, Nucl. Phys. B413 (1994) 481, hep-lat/9309005.

[27] ALPHA, M. Guagnelli, J. Heitger, R. Sommer and H. Wittig, hep-lat/9903040.

[28] ALPHA & UKQCD, J. Garden, J. Heitger, R. Sommer and H. Wittig, hep-lat/9906013.

[29] ALPHA, M. Lüscher, S. Sint, R. Sommer, P. Weisz and U. Wolff, Nucl. Phys. B491 (1997) 323, hep-lat/9609035.

[30] R. Sommer, Nucl. Phys. Proc. Suppl. 60A (1998) 279, hep-lat/9705026.

[31] ALPHA, K. Jansen and R. Sommer, Nucl. Phys. B530 (1998) 185, hep-lat/9803017.

[32] S.C. Eisenstat, SIAM J. Sci. Stat. Comput. 2 (1981) 1.

[33] M. Albanese et al., Comput. Phys. Commun. 45 (1987) 345.

[34] C. Battista et al., The APE-100 computer: 1. The architecture, in *Field theory, disorder and simulations*, edited by G. Parisi, Singapore, 1993, World Scientific.